International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 5, May2014)

Design and Implementation of High Speed Modified Booth Encoder Multiplier for Signed and Unsigned Numbers

Premakumar Waggi[1], Veeresh Pujari[2]
[1]M.tech student, VLSI Design and embedded systems Department, VTU RC Gulbarga, Karnataka, India
[2] Asst. Professor, VLSI Design and embedded systems Department, VTU RC Gulbarga, Karnataka, India

Abstract: **In this paper we design and implementation of modified booth encoder multiplier for both signed and unsigned numbers for 8 bit,16 bit,32 bit and 64 bits signed and unsigned numbers by using Radix-2,Radix- 4 and Radix-8 concepts. The implementation of project is done through Verilog coding and it is simulated on ISE Xilinx 12.1 platform. Modification is done in this project is both signed and unsigned number multiplication is performed by same multiplier and it is designed by Radix-8 algorithm and bit size of the number also increased. In this paper we analyze various parameters of multipliers such as area, speed, circuit complexity and time.**

Key Words: **Booth's multiplier, signed and unsigned numbers, partial products, radix-2, radix-4, radix-8.**

1. INTRODUCTION

Multiplication is a mathematical operation that at its simplest is an abbreviated process of adding an integer a specified number of times. Multiplication is the fundamental arithmetic operation important in several processors and digital signal processing systems. Multiplication of two k bit number needed multi operand addition process that can be realized in k cycles of shifting and addition with hardware, firmware or software.

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallograph y at Birkbeck College in Bloomsbury, London.
Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

Generally Multiplication contains three main steps: a) Recoding and generating partial products.
b) Minimizing the partial products by partial product reduction schemes (e.g Wallace tree) to two rows.
c) Adding the remaining two rows of partial products by using a carry-propagate adder (e.g., carry look ahead adder) to obtain [1][2].

This paper describes that in previous multiplier performs efficient multiplication operation such as it generatin g regular partial product and that can be reduced by various partial product reduction techniques due to which we can reduce the circuit complexity, area, power and delay of MBE multiplier. But disadvantage of this multiplier is it works only for signed number operations. But proposed modified-Booth algorithm is extensively used for high-speed multiplier circuits and it is work for both signed and unsigned number operations [5]. In this modified-Booth encoder multiplier we are using radix_2, radix_4 and Radix_8 algorithm for signed and unsigned number multiplication.

2. PROPOSED WORK

2.1 Modified Booth Multiplier

Booth encoding is a method of reducing the number of partial products required to produce the multiplication result. To achieve high-speed multiplication, algorithms using parallel counters like modified Booth algorithm has been proposed and used. This type of fast multiplier operates much faster than an array multiplier for longer operands because it's time to compute is proportional to the logarithm of the word length of operands. By recoding the numbers that are to be multiplied, Modified Booth multiplier allows for smaller, faster multiplication circuits. The number of partial products is reduced to half, by using the technique of Booth recoding [3],[4].

2.2 Radix-2 Booth Multiplier

Our multiplier is of the iterative Radix-2 Booth Multiplier type, implemented using asynchronous circuits [4, 5]. An iterative implementation was chosen, as opposed to a combinational array type, for higher area efficiency. A Booth implementation was chosen so as to uniformly handle signed as well as unsigned operands. However, a minor modification to the controller can easily transform our design into a simple (*i.e.* non-Booth) iterative multiplier.

A. Radix-2 Booth Multiplication Algorithm

Booth algorithm gives a procedure for multiplying binary integers in signed –2's complement representation. The booth algorithm with the following example:
Example: 2 ten × (–4) ten
0010 two × 1100 two
Making the Booth table

I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.
i.e., 0010-From 0 to 0 no change, 0 to 1 one change,
1 to 0 another change, and so there are two changes on this one,
1100-From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one. Therefore, multiplication of 2 × (–4), where 2 ten (0010 two) is the multiplicand and (–4) ten (1100two) is the multiplier.
II. Let X = 1100 (multiplier)
Let Y = 0010 (multiplicand)
Take the 2's complement of Y and call it –Y
–Y = 1110
III. Load the X value in the table.
IV. Load 0 for X-1 value it should be the previous first least significant bit of X
V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.
VI. Make four rows for each cycle; this is because we are multiplying four bits numbers.

B. Synthesis of Radix-2 Booth Multiplier

This is RTL Schematic of Radix-2 Booth Multiplier shown in Fig.1. Here first block is for finding the Multiplier between two given numbers and Second block is for the Multiplication process.

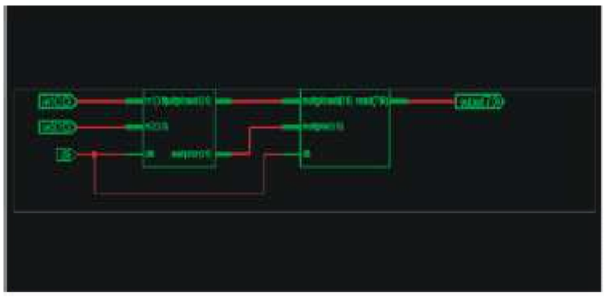


Fig.1 RTL Schematic of Radix–2 Booth Multiplier

2.3 Radix-4 Booth Multiplier

A. Radix-2 Booth Multiplication Algorithm

Radix-4 Booth algorithm which scan strings of three bits with the algorithm given below:
(1) Extend the sign bit 1 position if necessary to ensure that n is even.
(2) Append a 0 to the right of the LSB of the multiplier.
(3) According to the value of each vector, each Partial Product will be 0, +*y*, –*y*, +2*y* or –2*y*. The negative values of *y* are made by taking the 2's complement. The negative values of *y* are made by taking the 2's complement and Carry-look-ahead (CLA) fast adders are used. The multiplication of *y* is done by shifting *y* by one bit to the left. Thus, in any case, in designing n-bit parallel multipliers, only *n*/2 partial products are generated [5].
To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier.

Let us consider an example:
Multiplicand - (001011)2
Multiplier - (010011)2
First of all we will make group of three bits for Multiplier as shown in Fig.2.

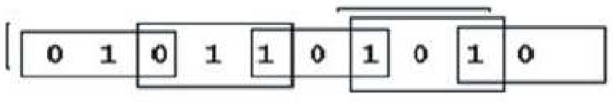


Fig.2 Grouping of bits for Multiplier.

Encoding for Radix-4 Booth Multiplier will be done according to the table given below:

Table1: Encoding of Radix-4 Booth Multiplier.

Block	*Partial Product*
000	0
001	1*Multiplicand
010	1*Multiplicand
011	2*Multiplicand
100	–2*Multiplicand
101	–1*Multiplicand
110	–1*Multiplicand
111	0

From the above table
(010)2 - 1
(001)2 - 1
(110)2 - (-1)
Therefore Multiplicand is multiplied with the three encoded digit which is 1, 1 and (–1).
(i) –1 * (001011)2 = (001011)2
And 1111 added with the result because of negative sign. Thus final answer of multiplication of (-1) is
(1111001011)2 {Negative term Sign Extended}
(*ii*) 1 * (001011)2 = (001011)2
(*iii*) 1 * (001011)2 = (001011)2
(*iv*) (00001)2 are added with these three resultants as an error correction for negation.
1 1 1 1 1 1 0 1 0 0 Negative term sign extended
0 0 1 0 1 1
0 0 1 0 1 1
0 0 0 0 1 Error Correction for Negation
0 0 1 1 0 1 0 0 0 1 Discarding the carried high bit

B. Synthesis of Radix-4 Booth Multiplier

Fig.3 shows detail diagram of RTL Schematic of Radix-4 Booth Multiplier which provide Output according to the Algorithm

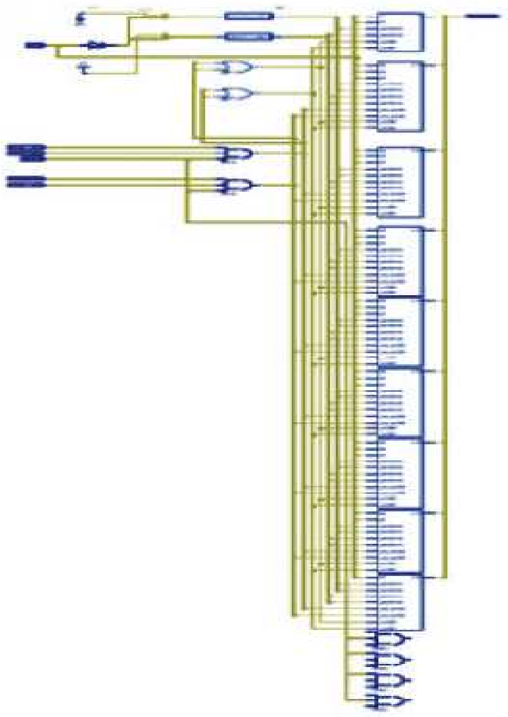


Fig.3 RTL schematic of Radix-4 Encoder Multiplier

2.4 Radix 8 Booth Multiplier

Booth's algorithm involves repeatedly adding one of two predetermined values to a product P, and then performing a Rightward arithmetic shift on P.Radix-8 Booth encoding is most often used to avoid variable size partial product arrays. Before designing Radix-8 BE, the multiplier has to be converted into a Radix-8 number by dividing them into four digits respectively according to Booth Encoder Table given afterwards[6]. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier.

S_u&Y7 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0 ‘0’ S

3. SIMULATION RESULT

3.1 Radix-2 Modified Booth Encoder Multiplier

Fig.4 shows Multiplication of two 8 bit signed numbers i.e n1=10011100(-100), n2=11110110(-10) P=n1*n2=10011100(-100)*11110110(-10)=0000001111101000(1000).

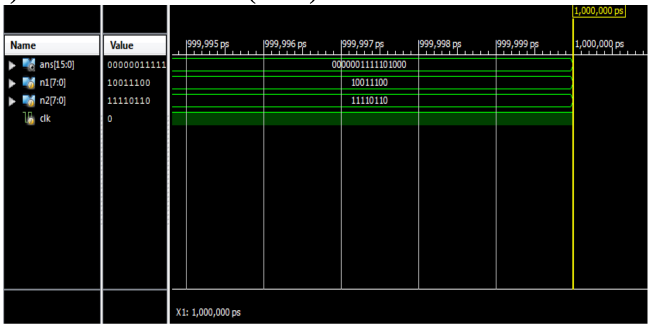

```
Name        Value            999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
ans[15:0]   00000011111      0000001111101000
n1[7:0]     10011100         10011100
n2[7:0]     11110110         11110110
clk         0
X1: 1,000,000 ps
```

Fig.4 Multiplication of two 8 bit signed numbers

Fig.5 shows Multiplication of two 8 bit unsigned numbers n1=00110010(50), n2=00011001(25) P=n1*n2=00110010(50)*00011001(25)=0000010011100010(1250).

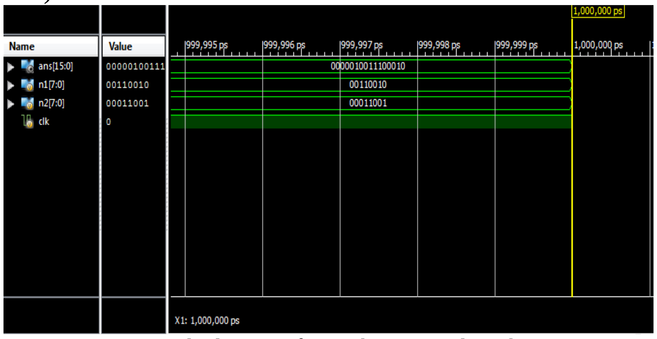

```
Name        Value            999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
ans[15:0]   00000100111      0000010011100010
n1[7:0]     00110010         00110010
n2[7:0]     00011001         00011001
clk         0
X1: 1,000,000 ps
```

Fig.5 Multiplication of two 8 bit unsigned numbers

3.2 Radix-4 Modified Booth Encoder Multiplier

Fig.6 shows Multiplication of two 16 bit signed numbers a=0000000000101100(44), b=1111111111100001(-31). P=a*b=0000000000101100 (44)* 1111111111100001 (-31)=11111111111111111111101010101100 (-1364).

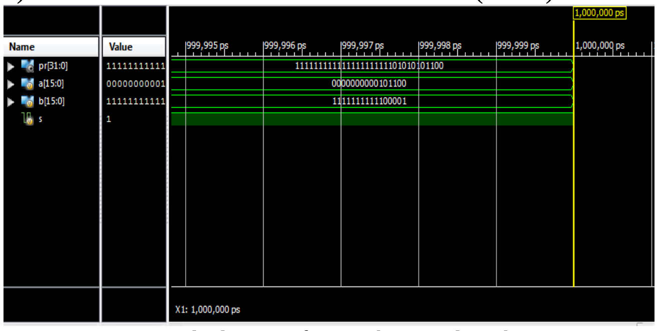

```
Name        Value            999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
pr[31:0]    11111111111      11111111111111111111101010101100
a[15:0]     00000000001      0000000000101100
b[15:0]     11111111111      1111111111100001
s           1
X1: 1,000,000 ps
```

Fig.6 Multiplication of two 16 bit signed numbers

Fig.7 shows Multiplication of two 16 bit unsigned numbers a=0000000001010101(85), b=0000000000110010 (50). P=a*b=0000000001010101 (85)* 0000000000110010 (50)=00000000000000000001000010011010 (4250).

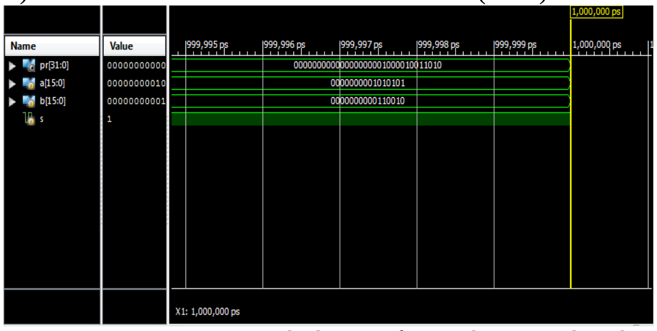
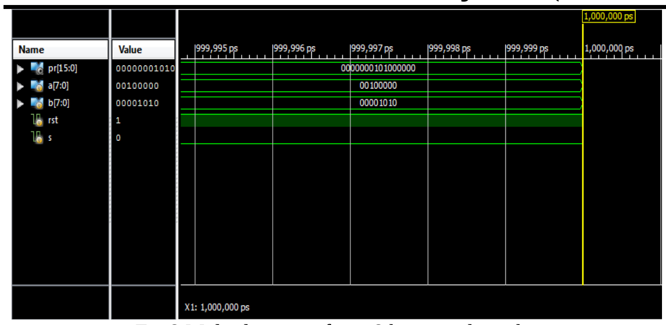

```
Name        Value            999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
pr[31:0]    00000000000      00000000000000000001000010011010
a[15:0]     00000000010      0000000001010101
b[15:0]     00000000001      0000000000110010
s           1
X1: 1,000,000 ps
```

Fig.7 Multiplication of two 16 bit unsigned numbers

3.3 Radix-8 Modified Booth Encoder Multiplier

Fig.8 shows Multiplication of two 8 bit signed numbers i.e a=00100000(32), b=00001010(10). P=a*b=00100000(32)*00001010(10)=0000000101000000(320).

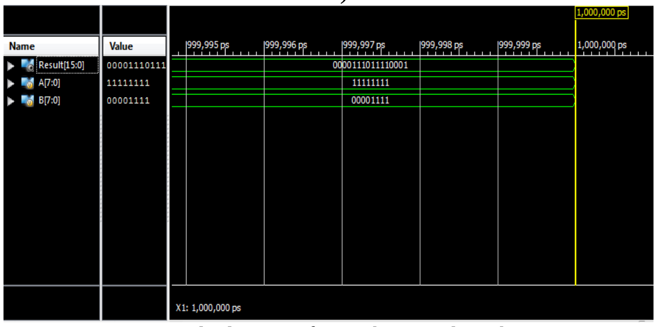


Name	Value
pr[15:0]	00000001010
a[7:0]	00100000
b[7:0]	00001010
rst	1
s	0

Fig.8 Multiplication of two 8 bit signed numbers

Fig.9 shows Multiplication of two 8 bit signed numbers i.e a=11111111(255), b=00001111(15).

P=a*b=11111111(255)*00001111(15)=0000111011110001(3825).



Name	Value
Result[15:0]	00001110111
A[7:0]	11111111
B[7:0]	00001111

Fig.9 Multiplication of two 8 bit signed numbers

4. CONCLUSION

By analyzing above results Modified booth encoder multiplier has been successfully implemented for both signed and unsigned numbers by using ISE Xilinx 12.1 software. Obtained simulation result waveform exactly matches with desired values of the multiplier. By using this methodology we got faster results of signed and unsigned numbers. Using different adder circuit we can still reduce the delay and complexity of the multiplier.

REFERENCES

[1]. Magnus Sjalander and Per Larson-Edefors.“The Case for HPM-Based Baugh-Woolley Multipliers,”Chalmers University of Technology, Sweden, March 2008.

[2]. Aswathy Sudhakar, and D. Gokila, “Run-Time Reconfigurable Pipelined Modified Baugh-Wooley Multipliers,” Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 223–235.

[3]. C R Baugh and B. A Wooley, “A two’s complement parallel array multiplication algorithm,” IEEE Transaction on Computers, Vol. 22, n0.12,pp 1045-1047, Dec.1973.

[4]. Shiann-Rong Kuang, *Member, IEEE*, Jiun-Ping Wang, and Cang-Yuan Guo” Modified Booth Multipliers With a Regular Partial Product Array” IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 56, No. 5, May 2009 1549-7747

[5]. *Ravindra P Rajput and M. N Shanmukha Swamy”* High speed Modified Booth Encoder multiplier for signed and unsigned numbers” , 2012 14th International Conference on Modelling and Simulation 978-0-7695-4682-7/12 © 2012 IEEE.

[6]. Minu Thomas “Design and Simulation of Radix-8 Booth Encoder Multiplier for Signed and Unsigned Numbers” IJSRD - Vol. 1, Issue 4, 2013 | ISSN (online): 2321-0613